Security Architectures for COTS based Distributed Systems

Pierre Bieber, Pierre Siron
ONERA-CERT
BP 4025, 2 avenue E. Belin
31055 Toulouse Cedex 4
France
Pierre.Bieber@cert.fr, Pierre.Siron@cert.fr

Abstract

The paper describes two experiments in the design of security architectures for distributed systems that are implemented with Commercial Off The Shelf components.We added security components to protect information exchanged in a Distributed Interactive Simulation environment. We added a role-based access control component to a Workflow tool implemented with CORBA technologies. The two experiments followed the same approach that includes four steps (threat analysis, security policy definition, selection of security components and architecture efficiency evaluation).

1 Introduction

Economical incentives are forcing the use of COTS (Commercial Off The Shelf) components in the design of complex distributed systems in the military sector. In this context, we are interested in securing COTS based systems.

The use of COTS components introduces several difficulties. First, we cannot use existing COTS components to enforce security because COTS components generally offer limited security services. Another difficulty is that we have to guarantee security even if we lack a detailed knowledge of how the components are implemented. COTS components developers often provide poor technical documentation. COTS component source code is generally not released with the notable exception of "open source software". Although this new breed of components is very promising we did not consider it in this paper because the systems we studied did not rely on free components. Finally, a common feature of COTS based system is that security is addressed when the development of the system is almost finished. Hence, the components used by the system can no longer be modified to guarantee security

Our job is to analyse the "software architecture" of a distributed system in order to extend it with components that guarantee security. By software architecture we mean a description of the components the system is made of and a description of how these components interact. Generally a description of how components are physically distributed over a network is available. But we need a more detailed description such as an object model that would list the classes of objects the system use and a set of scenarios that would explain how the objects interact.

In the following of this paper we describe our approach to design security architectures for COTS based distributed systems. We illustrate our approach with the security architectures we designed for two distributed systems. The first one is a Distributed Interactive Simulation environment. This system let interoperate simulations developed by various companies or military organisations. This kind of system is used to simulate new weapons or to conduct virtual manoeuvres. The second system is a collaborative workflow tools. This system let users locate relevant resources to perform a task. This system can be used to mechanise administrative tasks.

In the first part of the paper we explain the common steps in the design of a security architecture. Then we illustrate them on the Distributed Interactive Simulation and the Collaborative Workflow tool. Finally we discuss the similarities of these two security architectures and we describe the impact of the new compoents on original security architectures.

2 Design of Security Architectures

The first step is the analysis of the software architecture in order to find out what threats could be applied to the system. We consider two classes of threats:

- Attacks directed at the interaction between components. of a system. During their transit on the communication links between components (such as the Internet or a LAN) messages are subject to possible eavesdropping, or even worse blocking, replaying or forging.
- Attacks directed at the components. As components of the system were not developed with security requirements in mind, they could be used in order to disclose confidential data or in order to modify illicitly critical information.

We have to consider what attacks should really be taken into account. This involves assuming that some components are trusted not to perform attacks. For instance, some interaction links will be considered as secure because no component will try to listen to it. These assumptions generally remain unverified. One possibility would be to certify the components with respect to security evaluation criteria such as the ITSEC [15] or the new Common Criteria [12]. But the certification process is very expansive. So this option does not seem compatible with the cost-reduction imperatives that led us to rely on COTS components.

Paper presented at the RTO IST Symposium on "New Information Processing Techniques for Military Systems", held in Istanbul, Turkey, 9-11 October 2000, and published in RTO MP-049.

The second step is the definition of the security policy that lists the security objectives that should be satisfied. An obvious security objective is that all the attacks should be appropriately countered. So we have to define what information disclosure or modification are authorised with respect to the application under consideration. Although in previous papers, we have considered security policies in the context of multi-company co-operation where the main security objective is the protection of "Intellectual Property" and hence a confidentiality concern. In this paper we stress on applications where data has to be shared by civilian and military organisations. This involves both a confidentiality requirement: military private data should not be disclosed to civilian components and an integrity requirement: civilian components should not introduce erroneous information in military components.

The third step is the selection of components that enforce the security objectives that were selected during the previous step. For economical reasons we will try to use SCOTS (Security Components Off-The Shelf) when possible and try to limit the development of ad-hoc security components. SCOTS that protect the communication links implement security protocols such as Secure Socket Layer (SSL) [?] or Generic Security Service (GSSAPI) [7]. To implement access controls we can use SCOTS that perform IP packet filtering included in Firewall [17] toolkits such as Gauntlet from TIS.

The fourth and final step in the design of a security architecture is the evaluation of its efficiency. As stated previously, we think that it is unlikely that individual components of a COTS based distributed system will be certified. But, ITSEC evaluation principles could be applied in order to assess the assurance-efficiency of a security architecture. A completeness analysis could be conducted to see whether the threats are correctly by the selected security components. A consistency analysis could test whether the security components interact properly. Finally, an impact analysis could measure what is the impact of the new components on the original architecture.

3 Security Architecture for HLA/RTI

3.1 CERT HLA/RTI prototype architecture

ONERA/CERT has developed a prototype of distributed interactive simulation environment conforming to the HLA RTI standard (see [1] and [5]). In the following, we will use federate to denote an individual simulation and federation to denote a group of federates. Our prototype (see [2] and [3]) is made of several components: each federate interacts with a RTI Ambassador component (RTIA) and RTIA components interact with the RTI Gateway (RTIG) component. The RTI architecture is depicted by Figure 1.

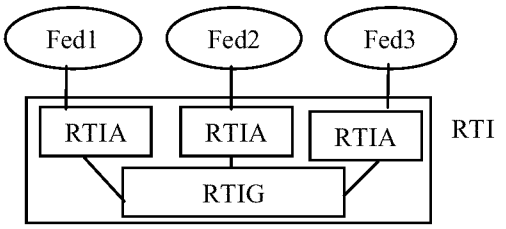

Figure 1. RTI architecture

The RTIA components are processes that exchange messages over the network, in particular with the RTIG process, via TCP (and UDP) sockets, in order to run the various distributed algorithms associated with the RTI services. RTIG is a centralisation point in the architecture. It uses the Federation Object Model (FOM) that describes the classes of data that federates can exchange during an execution. The RTIG records the identity of federates willing to publish data belonging to a class of the FOM or subscribe to a class of the FOM. The RTIG uses this information to forward messages in a multicast approach.

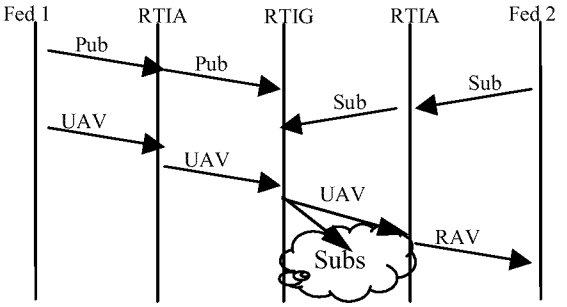

Figure 2. RTI data-transfer scenario

Figure 2 illustrates the message exchanges involved when federate 1 wants to inform other federates of the new value of an object. We suppose that in a previous step, federate 1 informed the federation that it was willing to publish values for that a class of objects in the FOM. An UAV (Update Value) message is sent to the RTIA and forwarded to the RTIG. The RTIG forwards this message to the RTIA of federates that subscribed to this class. If federate 2 has subscribed to this class, its ambassador will send it a RAV (Reflect Value) message as soon as the delivery condition holds.

3.2 Threat Analysis

We are considering a federation where simulations of both civilian and military organisation interoperate. We call FedM federates belonging to the military organisation and FedC the civilian federate.

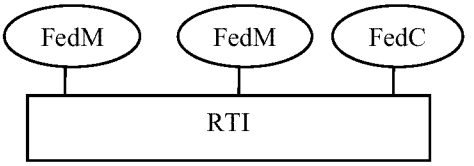

Figure 3. A dual federation

A detailed security analysis was performed, it is described in [3]. We distinguish two channels that Civilian federates can use to obtain Military information.

The first disclosure channel is related to attacks directed at the communication links between components of the RTI. It is likely that organisations will prefer that their federates run on hosts that belong to their local area network. These federates have to use a Wide Area Network such as the Internet to exchange messages with other components of the RTI. We assume that a federate and its RTIA process are executed on the same host so that we will not consider that the interaction link between them might be attacked. So we should only protect the communication link between a RTIA and the RTIG.

The second channel is the leak of information via the RTI services. A malicious federate could try to use its services in order to gain knowledge about a private object. One insecure scenario occurs when federate FedC subscribes to a class that happens to be regarded as private by the Military. The normal behaviour of RTI will be to forward values of objects in the private class to federate FedC. So HLA/RTI data transfer services could be used to disclose confidential data. We suppose that the RTIG process is under the control of a third party that is trusted by all the organisations. For instance, this third party could be a public organisation running a simulation facility. So the RTIG will not disclose private date intentionally. An organisation may trust the federate component it has written, it might also trust components of the RTI such as its RTIA or RTIG. But it would certainly not trust federate components developed by other organisations, so we should forbid a federate from one organisation to use the RTI services in order to learn private information belonging to another organisation.

3.3 Security Policy

In order to limit data exchanged using HLA/RTI data transfer services such as Update Value / Reflect Value, we propose to associate a security label with objects and federates of the federation. The RTI will control the messages according to the security labels of the object and of the federate. Security labels we have considered are PrivateMil and Public. PrivateMil dominates security label Public.

The description of the federation must be completed with Federation Execution Data that include security label information. Figure 4 gives an example of security label association. In this federation, the class Aircraft is public whereas its sub-class Fighter is regarded as Private by the Mil organisation. We consider two federates: one modelling an air traffic controller that has security label Public and another one that models a military controller that has security label PrivateMil.

```
(fed
(objects
(class "Aircraft"
      (sec_level "Public")
      (attribute "Position")
    (class "Fighter"
      (sec_level "PrivateMil")
      (attribute "WeaponLoad")
...)))
(federate           "AirTrafficControl"
"Public")
(federate "MilControl" "PrivateMil")
...)
```

Figure 4: Federation security labels

The RTI should allow the Air Traffic Controller federate to observe the current position of a fighter but the RTI should not authorize this federate to know the current status of the weapon loaded on the fighter.

3.4 Selection of Security Components

With regard to the first threat, the security function needed should build a secure association between the RTIA and the RTIG. This association should guarantee authentication and confidentiality of the communication. The second threat can be resolved by adding access controls within the RTI services that should restrict the message exchanged between federates of two companies.

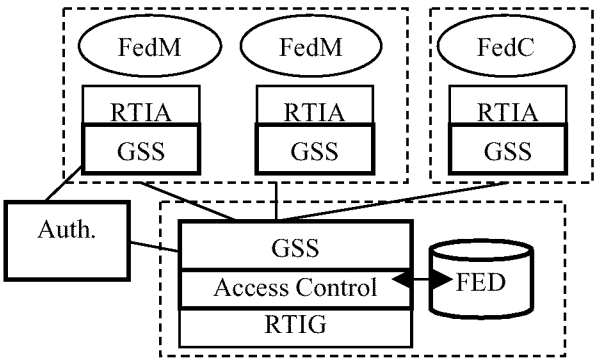

Figure 4. HLA/RTI Security Architecture

3.4.1 GSS-API security services

In a WAN context, protecting communication links can be done by using cryptographic techniques. Rather than developing a new cryptographic protocol, we selected the Generic Security Services Application Programming Interface (GSS-API) [7]. It is an Internet Engineering Task Force standard that defines cryptographic services that are useful to secure a client-server application. The services include the management of encryption keys, the distribution of shared key or using distributed keys to enforce the confidentiality, authentication or integrity of exchanged messages.

The GSS-API interface hides the details of the underlying security mechanism leading to better application portability. In particular, this would allow changing the underlying security mechanism if a security flaw is encountered in it. Several popular security protocols offer a GSS-API interface such as Kerberos [8] or SESAME [9].

We used the GSS-API implementation developed at DTSC in Australia [10]. GSS-API was integrated to the RTIA and RTIG processes to secure their communication. We extended the Socket class that is used to exchange any messages within RTI. Ambassadors of remote federates will use sub-class SecureSocket that includes the appropriate calls to the GSS-API services whereas Ambassadors of local federates will use class Socket.

The SecureSocket class hides several steps that should be performed to protect a communication link. Prior to the execution of a federation, every federate ambassador has to contact the Authentication server that will provide the ambassador with credentials (an encryption key tied together with the identity of the federate and a date). When a federate wants to join a federation its RTIA has to contact the RTIG and authenticate itself using the credentials. This involves several exchanges of messages between RTIA and RTIG. If this step succeeds, a security association is created between RTIA and RTIG (a session key is distributed to both processes). Then, RTIA and RTIG can protect the HLA/RTI messages they exchange by using cryptographic functions. For instance, if integrity has to be enforced an elaborate message digest (the MD5 algorithm is used) will be appended to each message, if confidentiality has to be enforced all exchanged messages will be encrypted (the DES algorithm is used) with the distributed session key.

3.4.2 Publish/subscribe access controls

Publication and subscription messages are controlled rather than data-transfer messages. A federate will be authorised to subscribe to a class if its security label dominates or is equal to the security label of the requested class.

This control is performed by the RTIG because, in our architecture, this component is already in charge of recording the publication and subscription. So the RTIG will now check the security labels of the federate and of the class whenever this federate has sent a subscription message for this class. The RTIG will record for each published class a list of authorised subscribers. As the RTIG transmits Update Value messages only to authorised subscriber RTIA, a federate from one company will never receive Reflect Value messages for a private object of another company because its subscription request are blocked by the security label control in the RTIG.

The RTIG is extended with new services that manage the security labels. These services use the Federation Execution Data to associate security labels with federates as they join a federation, and with the classes in the FOM. Furthermore, a function comparing two security labels has been implemented (this function is quite independnet from the security labels used in the security policy), it is used to check whether a subscription message should be discarded. In this case, we generate an exception.

4 Security Architecture for CIDRIA

4.1 CIDRIA Architecture

The CIDRIA workflow tool was developed by France Telecom/CNET. It is described in [11]. CIDRIA is implemented as a CORBA [16] server that handles various objects : *Agent* objects that represent users or resources and a *Cooperator* object that supervises Agent objects.

One benefit of using CORBA to implement CIDRIA is that each class is associated with an IDL interface that lists the methods it offers. In the following, we illustrate how these methods can be used. For that purpose, we suppose that CIDRIA is used to implement a flight planning system. Pilots would use this system to prepare their flight mission. A map provider resource called IGN is connected to CIDRIA as well as a commander that will give the mission goal to the pilots. We suppose that the map provider is a civilian organisation whereas pilots and commanders belong to a military organisation

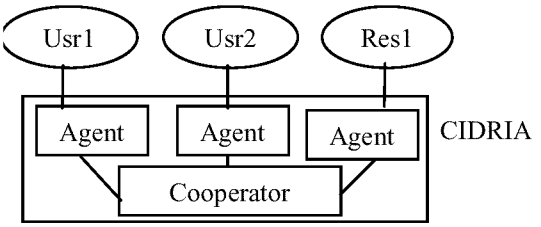

Figure 5. CIDRIA architecture

First of all, when a resource or a user client logs on CIDRIA, it invokes method `connect` of object Cooperator. This creates an object of class Agent, then the client interacts exclusively with this object. When a pilot wants to prepare a mission, it will invoke method `add_request` of its agent with parameter `map` to request a map. The agent submits this request to the cooperator that tries to locate an agent that could provide a map.

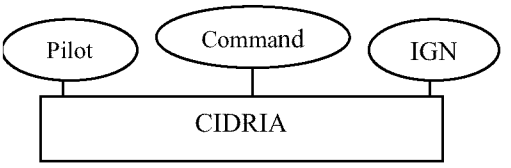

Figure 6. Flight planning application

The Cooperator is able to locate resource IGN if it previously registered its competence on the topic map using method `add_competence` of its agent. Then the request is sent to IGN that can decide to answer it (and provide the requested map) or not using methods

`reply_request` or `reject_request` of its agent. If the request is rejected the Cooperator will try to locate another resource providing maps.

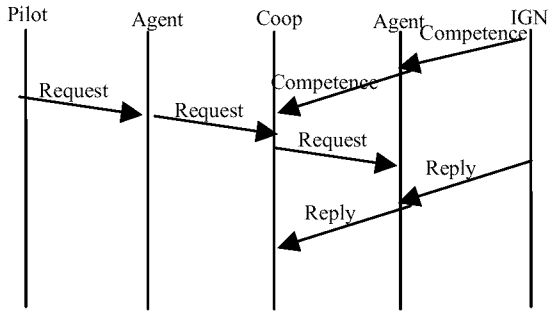

Figure 7. Request resolution scenario

4.2 Threat Analysis

As in the case of HLA/RTI we consider two disclosure channels. The first disclosure channel is related to attacks directed at the communication link between CIDRIA and its clients. In the case of CIDRIA, messages exchanged between CIDRIA and the clients contain requests, responses or object localisation information. Hence, these attacks could allow an intruder to invoke any method of CIDRIA. We should protect the communication link between a client and the CIDRIA.

The second channel is the leak of information via CIDRIA services. We suppose that the CIDRIA server is operated by an organisation that does not trust its clients to behave correctly. A malicious client could try to use its services in order to gain knowledge about a private data or to provide erroneous information. For instance, a malicious client could claim to be a commander using method `add_competence` then the normal behaviour of CIDRIA is to forward the pilot requests for mission goals. So a malicious client could reply to these requests and provide erroneous mission goals to the pilots. A malicious client could also send a request for mission goals using method `add_request` with parameter `goal`. CIDRIA would forward the request to the commander. If the commander replies to the request, then a malicious client could learn what is the mission goal, which is likely to be a confidential information.

We should control that a client is authorised to claim a competence on a topic. For instance, a civilian client is not authorised to claim its competence on mission goals. We should also control that a client is authorised to request data on a topic. For instance, a civilian client is not authorised to ask for mission goals.

4.3 CIDRIA Security Policy

We selected "Role-based Access control" RBAC policy (see [13] and [14]) to describe CIDRIA security objectives. In a RBAC policy, a set of roles is associated with each user according to the tasks this user should perform within the organisation. A role is set of operations that a user playing this role is authorised to perform.

In order to define CIDRIA roles we consider that CIDRIA operations are methods that appear in the interface of classes Cooperator or Agent with their parameters. Example of methods is `add_competence` or `add_request`, and examples of parameters are: `maps` or `goals`. We define the following roles: Pilot, Commander and MapProvider. A Pilot is authorised to invoke method `add_request` with parameter `maps` or `goals`. A commander is authorised to invoke method `add_competence` with parameter `goals`. And a Map Provider is authorised to invoke method `add_competence` with parameter `maps`.

A role access control component checks that a client is authorised to play the role it selects when it connects to CIDRIA. And the access control component should control that the method invoked by a client is authorised according to the role it is playing.

4.4 Selection of security components

The threat analysis showed that components that protect communications between a client and CIDRIA should be added. We could select the GSS-API component just as in HLA/RTI. So we do not detail the protection of communication links for CIDRIA and we focus on access control components.

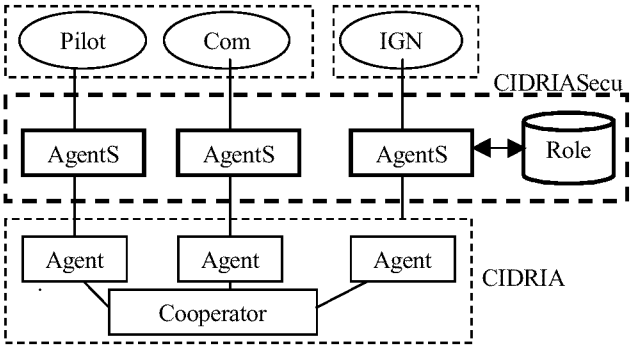

Figure 8. Secure Components added to CIDRIA

4.4.1 Role-based Access Control Server

The CORBA standardisation body has defined a set of security services that include access-control. We decided not to use it because these services are not currently available in the most common CORBA implementations. Instead we decided to define a new server called CIDRIASecu that manages two new classes CooperatorS and AgentS. The interfaces of these classes are similar to class Cooperator and class Agent interfaces. Our goal is that clients invoke methods of the new CIDRIASecu component instead of CIDRIA server methods. When a client invokes a method of CIDRIASecu, this method first checks that the method with its parameters is an

authorised operation according to the role that the client is playing. If the operation is authorised then the synonymous method of server CIDRIA is called otherwise an exception is generated.

The access controls performed by the methods of class AgentSecu are based on the description of the roles. A Prolog database contains predicates that describe authorised operations for a role, and authorised roles for a client. The prolog database is encapsulated within a C++ object that can be used by all the AgentS components.

4.4.2 Border Access Control: Firewall and Wonderwall

In the last section, we made the assumption that clients would invoke the methods of server CIDRIASecu rather than the methods of CIDRIA. Of course a malicious client would certainly try to bypass the controls performed by CIDRIASecu and try to invoke directly the methods of CIDRIA. To avoid this situation, we propose that CIDRIA and CIDRIASecu servers run in a protected area called an enclave. Inside the enclave all the components are trusted and outside the enclave the components such as the clients are supposed to be malicious.

So we should add functions that protect the border of the enclave (i.e. the interactions between clients and the components inside the enclave should controlled). We want to forbid clients to invoke methods of server CIDRIA. For that purpose, we used a SCOTS called Wonderwall [18] (produced by IONA). Wonderwall controls what objects are accessed on a server and what methods are invoked on these objects. In our security architecture, Wonderwall will allow accesses to objects in classes AgentS and CooperatorS and it will forbid accesses to objects in classes Agent and Cooperator. We added a Firewall that filters any communication whose destination is not the Wonderwall.

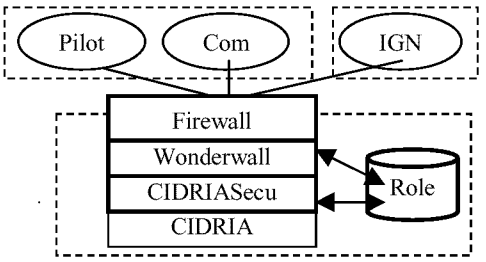

Figure 9. CIDRIA Security Architecture

The previous figure shows the interaction between a client, CIDRIASecu and CIDRIA. A request from the client proceeding from Internet goes through the firewall then Wonderwall filters it. If the request is authorised, Wonderwall forwards it to CIDRIASecu. If the request passes the role controls, CIDRIASecu forwards it to CIDRIA. The response to this request takes the same path in the reverse way.

5 Conclusion: Security Architecture Evaluation

In this concluding section, we try to evaluate efficiency of the two architectures we have designed. We first explain the similarities between both security architectures then we evaluate their impact on the original architectures.

5.1 Similarities of the two security architectures

The two systems we have considered could be regarded as rather simple instances of COTS based system. Both systems were designed in an object-oriented way, hence we could use an appropriate description of their software architecture. Furthermore, the systems were developed by partners (ourselves for HLA/RTI and France Telecom/CNET for CIDRIA) so we could get all the details we needed to know on the behaviour of the components. Nevertheless, we applied our approach to secure systems according to the rules stated in the introduction: we did not modify any existing components nor did we use our knowledge of the components implementation. We mainly added new components. We used several SCOTS: a firewall, GSS-API security protocol and the Wonderwall CORBA messages filtering tool. So we believe, that our approach could be valid in less forgiving cases. For instance, our approach would still be valid if we totally ignore the source code of components.
For both systems we had to develop ad-hoc security components that control whether the services offered by the distributed system are correctly used. It was not possible to use SCOTS because these security components depend on the application security policy we want to consider and on the services of the distributed system we want to protect.

5.2 Impact

Once the security architecture was designed and implemented we were able to study the impact of the new components we added on the overall behaviour of the system. As the two applications have quite different purposes we did not study the security impact in the same way. HLA/RTI is rather focused on real-rime performances whereas CIDRIA is an information system tool where ease of management matters.

To assess the impact of security components on the real-time performances of HLA/RTI we used a toy federation modelling three aircraft trajectories. We compared the number of simulation steps performed by the simulation with and without security components.
As access control is only performed during the subscription stage of a federation, this component has no influence on the main stage of a federation that generally involves a lot of data-transfer message exchanges. So we have not observed any significant decrease of real-time performances when we added access controls.

The first experiments we made showed a 25% decrease of the performances when GSS-API is used to protect message integrity and much more when confidentiality is protected. The federates use very simple trajectory computation and communicate a lot, so the performance of SecureSocket functions have a huge impact on the overall performance of the federation. So we expect that performance decrease would not be as great in a realistic simulation with federates performing more complex computations.

The CIDRIA tool has a complex administration procedure that should be performed before clients connect to the server. An administrator client should invoke CIDRIA methods in order to create or destroy agents, to add or remove request topics. The CIDRIASecu server we added should also be administered in this way, furthermore the server needs a description of the roles. So we had to introduce a security officer that invokes CIDRIASecu methods that add, modify or remove a role, an operation or an identity. This security officer is also in charge of creating instances of AgentS objects and creating request topics that are managed by CIDRIASecu. The resulting administrative procedure is cumbersome so we decided to simplify. When a role is defined by the security officer, an instance of Agent and Agent S are automatically created. Similarly when operation is defined with a new request topic it is automatically added in CIDRIA and CIDRIASecu.

6 Acknowledgements

The study of HLA/RTI security architecture was funded by DGA/STTC. The study of CIDRIA security architecture was partially funded by France Telecom/CNET contract CTI n°97IB552.

7 References

[1] Department of Defense, "High Level Architecture Interface Specification, Version 1.3 Draft 7", January 1998.

[2] P. Siron, "Design and Implementation of a HLA RTI Prototype at ONERA", the Fall Simulation Interoperability Workshop, 1998.

[3] P. Bieber, J. Cazin, P. Siron, G. Zanon "Security extensions to ONERA HLA RTI Prototype", the Fall Simulation Interoperability Workshop, 1998.

[5] DMSO, "HLA/RTI web page", http://hla.dmso.mil

[7] J. Linn, "Generic Security Service Application Programming Interface", Internet RFC 2078, January 1997.

[8] J.Steiner, B. Neuman, J. Schiller, "Kerberos: An Authentication Service for Open Network Systems", proceedings of The USENIX Winter Conference, February 1988.

[9] P. Kaijser, J. Parker, D. Pinkas, "SESAME: The solution to security for Open Distributed Systems", Computer Communications, July 1994.

[10] D.P. Barton, L.J. O'Connor, "Implementing Generic Security Services in a Distributed Environment", Technical Report, CRC for Distributed Technology, Brisbane, Australia, April 1995.

[11] Bruno Dillenseger, François Bourdon, « Modélisation de la coopération et de la synchronisation dans les systèmes d'information – Une expérience de Workflow basée sur les nouvelles technologies », Calculateurs parallèles, volume 9, n 2, 1997.

[12] Common Criteria, http://csrc.nist.gov/cc

[13] Ravi Sandhu, Edward Coyne, Hal Feinstein, Charles Youman, « Role-based Access Control Models », Computer, February 1996, IEEE Computer Society Press,.

[14] D. Ferraiolo, J. Cugini, R. Kuhn, « Role Based Access Control: Features and Motivations », Proceedings 10th Annual Computer Security Applications Conference, IEEE Computer Society Press, 1994. http://waltz.ncsl.nist.gov/rbac/rbac/newpaper/rbac.html

[15] ITSEC, Information Technology Security Evaluation Criteria, Union Européenne, 1991.

[16] OMG, "CORBA Services". http://www.omg.org

[17]D. Brent Chapman, Elizabeth D. Zwicky, « La sécurité sur Internet – Firewall », Editions O'Reilly International Thomson, 1996.

[18] IONA, "Wonderwall", http://www.iona.com/products/orbix/wonderwall.html

This page has been deliberately left blank

Page intentionnellement blanche